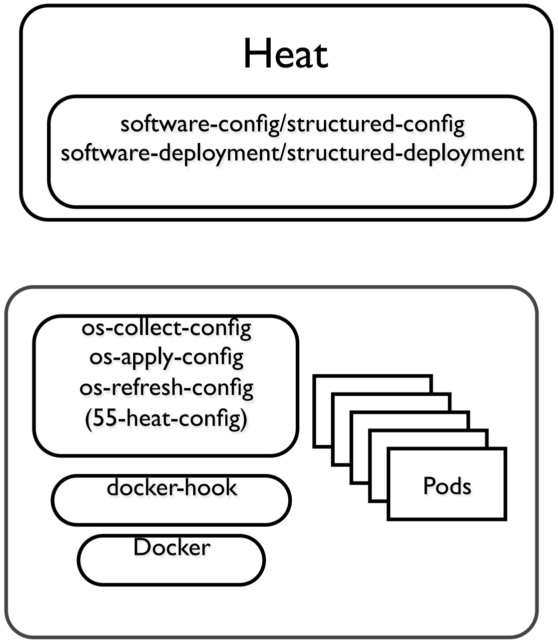
Proposed Implementation

https://review.openstack.org/#/c/128182/



Heat

software-config/structured-config
software-deployment/structured-deployment

os-collect-config
os-apply-config
os-refresh-config
(55-heat-config)

docker-hook

Docker

Pods

```
manifest:
    version: v1beta2
    containers:
     - name: apache_container
       image: fedora/apache
       command: [/run-apache.sh]
       volumeMounts:
        - name: test_volume
          mountPath: /log
          readOnly: true
       ports:
        - name: http_port
          containerPort: 80
          hostPort: 8000
          protocol: tcp
       env:
        - name: GOPATH
          value: /usr/bin
     - name: redis_container
       image: dockerfile/redis
       cpu: 100
       volumeMounts:
        - name: test_volume
          mountPath: /log
          readOnly: false
       ports:
        - name: http_port
          containerPort: 6379
          hostPort: 6379
          protocol: tcp
       env:
        - name: GOPATH
          value: /usr/bin
    volumes:
     - name: test_volume
```

Implementation Details:

1. Uses docker-py with sofware-config/deployment
2. Uses google container manifest(~ k8s POD)
3. pod_name - software_deployment ‘name’ property
container_name - <pod_name>.<given_container_name>
4. Teardown pod (all containers for a pod before create and update, not the best solution!
5. OUTPUT contains complete container_info(docker inspect) for all containers.
https://review.openstack.org/#/c/128182/

Limitations:

No schema validation
Validation of port and volume conflicts
Validation whether individual container config has changed

Future Work:

1. Support private repository/registry
2. Implement schema validation for container-config
(stevebaker) generic - separate blueprint

Some K8S Background

POD:

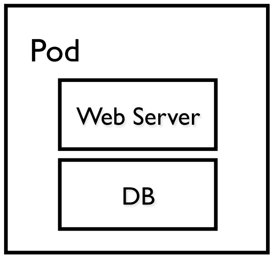
A pod is a relatively tightly coupled group of containers that are scheduled onto the same host. It models an application-specific "virtual host" in a containerized environment. Pods serve as units of scheduling, deployment, and horizontal scaling/replication.

KUBELET:

Kubelet, a k8s component, is a container agent that runs on nodes and works with container manifests. A container manifest is a YAML/JSON configuration that describes a pod. Kubelet agent running on each node takes a manifests (describing multiple pods) that can be provided in various (file/http url/etcd) mechanisms and ensures that the containers described in those manifests are started and continue running

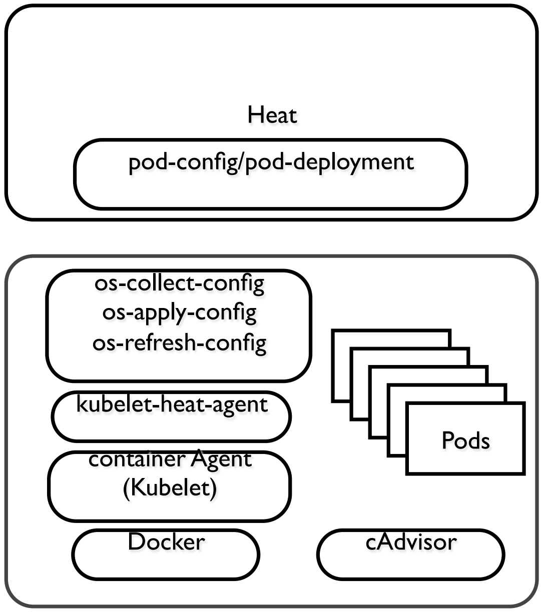
POD Operations Supported (GCE/k8s):

CREATE
VIEW
DELETE



Behavior	Benefits
Group of Containers	Reuse across environments
Settings in a Template	Repeatable Manageable

Alternate Approach with Kubelet



Discussion Points/Issues:

1. Kubelet pod management is non blocking REST API Calls for CREATE/UPDATE/DELETE

2. Kubelet provides self healing of a POD depending on the POD ‘restartPolicy’ defined in the manifest.

2. POD UPDATE not fully supported yet with k8s/kubelet
https://github.com/GoogleCloudPlatform/kubernetes/issues/1712)

3. pod-config/pod-deployment resources (may be subclass of software-config) and implement ‘config’ validation

4. kubelet-heat-agent(may be the hook itself) wraps the Kubelet Agent functionality and provides some additional polling for the status of the pods and containers and signals heat for the resource status - Use the zaqar queue?

5. No python client library for leveraging kubelet